**1. Array Representation**

**Description:**  
• Arrays are a linear data structure where elements are stored in **contiguous memory locations**.  
• Each element is accessed directly using its **index** (starting from 0).  
• All elements must be of the **same data type**.  
• The memory address of the i-th element is calculated as:  
  Base Address + i × Size of Data Type

**Advantages:**  
• **Fast access** using index — O(1) time complexity.  
• **Simple structure** and easy to implement.  
• **Efficient traversal** through loops.  
• Best suited for scenarios where the size is fixed and known.

**4. Analysis**

**Time Complexity of Operations**

| **Operation** | **Description** | **Time Complexity** |
| --- | --- | --- |
| **Add (at end)** | Insert at the last index | O(1) |
| **Add (at index)** | Insert at a specific index (requires shifting) | O(n) |
| **Search** | Linear search through elements | O(n) |
| **Traverse** | Visit all elements once | O(n) |
| **Delete (by index)** | Remove element (shift others to fill gap) | O(n) |

**Limitations of Arrays and When to Use Them**

| **Limitation** | **Explanation** |
| --- | --- |
| **Fixed size** | Array size must be known at creation; cannot grow dynamically. |
| **Inefficient insert/delete** | Requires shifting elements which is costly for large arrays. |
| **Wasted memory or overflow** | Risk of memory waste if over-allocated, or overflow if underestimated. |
| **Homogeneous data** | Cannot store mixed types (unless using structures/objects). |

**When to Use Arrays**

| **Situation** | **Recommended Use** |
| --- | --- |
| Fixed number of employee records | Use Arrays |
| Frequent direct access by index | Use Arrays |
| Need for dynamic growth or flexibility | Consider Linked List / ArrayList instead |
| Many insertions/deletions in the middle | Use more dynamic data structures like Linked Lists |